440-TP-010-001

DADS MR-AFS Proof of Concept Results for the ECS Project

Technical Paper

Technical Paper—Not intended for formal review or government approval.

February 1995

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

William Neville /s/ 2/28/95

Will Neville, Software Engineer Date
EOSDIS Core System Project

SUBMITTED BY

Stephen Fox /s/ 3/18/95

Steve Fox, SDPS Office Manager Date
EOSDIS Core System Project

Hughes Applied Information Systems
Landover, Maryland

This page intentionally left blank.

Preface

This document contains the Multi Resident (MR) Andrew File System (AFS) Proof of Concept (PCON) Report. This document is submitted as required by the ECS Statement of Work, Section 3.3.3.3, and does not require Government approval. The results of this proof of concept will also be documented in the Prototyping and Studies Final Report (DID 333/DV3).

For additional technical information pertaining to this Data Server PCON, contact Tom Smith or Will Neville, SDPS-DADS, at (301) 925-0647 / (301) 925-0658 or on email at tsmith@eos.hitc.com / wneville@eos.hitc.com.

Questions concerning the control or distribution of this document should be addressed to:

Data Management Office
The ECS Project Office
Hughes Applied Information Systems
1616 McCormick Dr.
Landover, MD 20785

This page intentionally left blank.

Contents

5. Conclusions

6. References

Figures

Appendix A - Prototype Data Points

Abbreviations and Acronyms

This page intentionally left blank.

1. Introduction

1.1 Identification

This Data Server PCON Report is prepared for the Earth Observing System Data and Information Systems (EOSDIS) Core System (ECS) project, contract number NAS5-60000.

1.2 Summary

This PCON has shown that an operating system such as AFS is a viable solution in meeting some Science Data Processing Segment (SDPS) requirements. The PCON has also shown that MR-AFS will support shared Redundant Arrays of Inexpensive Disks (RAID) storage devices.

1.3 Purpose

This PCON focused on a number of architectural and data management issues. First, it determined the applicability and viability of an operating environment such as AFS in meeting SDPS data requirements. AFS was the logical place to begin the PCON effort because it is an existing commercial product with a large user base, and it forms the core of the Pittsburgh Supercomputing Center's Multi Resident extensions.

A second issue was to establish whether MR-AFS will support shared RAID storage devices for multiple hosts. A capability for concurrent read/write file access would be prove invaluable for supporting simultaneous read/write staging access associated with the Ingest, Planning and Data Processing and Data Server subsystems at each Distributed Active Archive Center (DAAC).

1.4 Approach

The tests used in the PCON were designed to lockstep functionality and performance as the evaluation progressed. Each phase of testing built on functionality and performance measurements from previous tests. These related tests were designated as a "series". During each series of tests the hardware and software system configuration was held constant while the performance and functionality tests were performed. A standard test suite of files was used to collect the performance data.

1.5 Parent Document

February 1993 ECS Statement of Work

1.6 Applicable Documents

The following documents are applicable to this document:

193-707-PP1-002	ECS Prototype Results Review, submitted December 1993
193-216-SE1-001	ECS Requirements Specification, submitted February 1994
193-317-DV1-001	ECS Prototyping and Studies Plan, submitted May 1993
193-318-DV3-005	ECS Prototype and Studies Progress Report, submitted November 1993

2. Hardware Configuration

2.1 Hardware Configuration

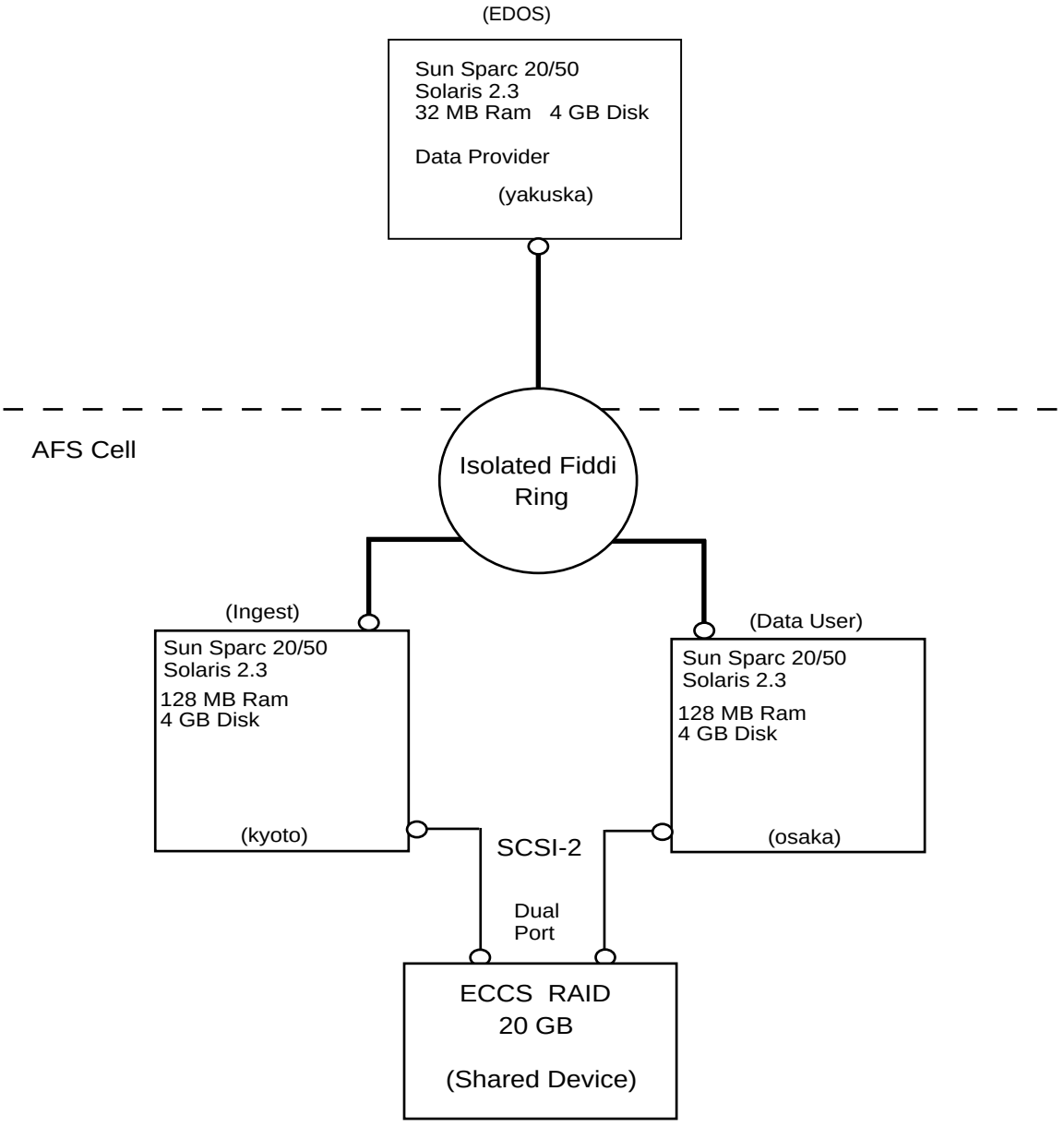

Figure 2.1-1. MR AFS Proof of Concept Hardware Configuration

This page intentionally left blank.

3. Andrew File System

3.1 Product Summary - AFS 3.3a

AFS is a distributed file system originally developed at Carnegie Mellon University, and first released commercially in early 1990. AFS allows multiple users to share and access files stored in a network of computers. Files may reside on many different machines but are available to users, if the user holds the requisite file permissions, on every machine. AFS stores files on certain machines in the network, called *file server machines*. These machines also provide the delivery service for the files to the other machines in the network called *client machines*. All of the machines that run either the AFS server software or the AFS client software are encompassed in an AFS cell. AFS allows the local file space for all the machines in a cell to be combined into a global uniform name space that makes file access transparent. Thus each user on every "in cell" machine sees the same path names. AFS groups files into *logical volumes* for administrative and performance management reasons. A volume is used to group related files and store them together in an AFS partition. Volumes can be moved from one partition to another transparent to the user. Because of the distributed nature of AFS, Kerberos authentication services are used for file security. Servers, clients and users all must prove their identities in order to be able to exchange information. Users are required to log into the AFS system by way of user name and password information before they can gain access to files stored in AFS. Users may also control access to files and directories that they own by way of access control lists.

3.2 Product Summary - MR-AFS 3.3a

The MR (Multi-Resident) extensions to the AFS environment were developed at Pittsburgh Super Computing Center (PSC). MR-AFS is based on AFS but is optimzed to efficeintly support mass storage systems by providing services such as automatic data migration across various storage media, multiple file residencies, and shared storage devices. These allows MR-AFS to provide an "infinite" AFS storage space. MR-AFS also supports dynamic hierarchical structures of storage systems by providing a system of prioritized access. Parameters are tunable to allow for maximum and minimum file sizes across various storage devices. MR-AFS volumes can exceed the 2 gigabyte maximum partition size imposed by the standard Unix file system because of the storage systems that it supports. Files within a volume can also reside on more than one type of media. When accessing a file that is on a shared storage device, the data path will not be through the server machine but directly to client (assuming the client has direct access to the shared device.) This allows for faster transfers with one less "data hop" to go through to get to the client machine.

This page intentionally left blank.

4. Proof of Concept Implementation

4.1 Proof of Concept Design

The design of the PCON emphasized functionality. Performance data was collected as a means of comparing divergent network and architectural configurations. The test suite consisted of 3 file groups of the following sizes: 1KB, 5KB, 10KB, 50KB, 100KB, 250KB, 500KB, 1MB, 5MB, 10MB, 50MB, 100MB, 250MB, 500MB, 1GB.

4.2 Evaluation Method / Criteria

This effort's success or failure was evaluated by applying the following criteria to the tests that were performed.

1. The AFS operating environment will be deemed viable in meeting SDPS requirements if, at a minimum, it demonstrates the following capabilities:

    A. Security and access control lists allow successful differentiation between the authorized and unauthorized users for the seven levels of access privileges provided by the product.

    B. There is no loss of continuity due to volume replication: data is not corrupted; the access from multiple points is transparent.

    C. Scalability of up to 2GB is demonstrated for file retrieval and storage caching. Speed and performance of caching are of essence.

2. The MR-AFS will be deemed viable in meeting SDPS requirements, if, at a minimum, it demonstrates the following capabilities:

    A. After duplication of all the tests performed on AFS, for MR-AFS there are no defects found in the performance of MR-AFS due to the changes associated with multi-residence.

    B. Allows multiple hosts (2 in the PCON configuration) to simultaneously share for reading and writing a large "free pool" of RAID space without data loss or significant performance degradation.

4.3 Test Cases/Procedures

4.3.1 AFS In-Cell Functional/Performance Series

Series Purposes:

- Establish AFS functional/performance benchmarks within cell
- Initial AFS cell administration and setup experience

Series Config:

- AFS Server on Kyoto
- AFS Client on Osaka
- ECCS RAID disk used by Kyoto
- Normal user and administrator accounts

Series Tests Performed:

a) The test suite of files was copied from Osaka to the RAID using the Unix cp command. The transfer and deletion times were recorded. The graph shows a maximum transfer rate of about 250KB/sec with declining performance as the size of the file increases. The transfer times are in Figure 4.4-1 of the Performance Results section of this paper and the deletion times are in Figure 4.4-2. The deletion results show a maximum rate of 5 MB/sec.

b) The following AFS rights and privledges were toggled and verified for proper user and administrative access to directories. There were no abnormal results.

   - LOOKUP
   - INSERT
   - DELETE
   - ADMINISTER

c) The following AFS rights and privledges were toggled and verified for proper user and administrative access to files. There were no abnormal results.

   - READ
   - WRITE
   - LOCK

d) The test suite was generated from Osaka to the RAID using a file generation tool. The creation times of theses files were recorded. The results of this test are in Figure 4.4-3 of the Performance Results section of this paper. The results show a maximum file generation rate of about 220KB/sec with minimal dropoff as file size increases.

e) The test suite was copied to the RAID device and the files were FTPed back to Osaka from a read-only copy of the volume. The transfer rates were recorded. The results of this test are in Figure 4.4-4 of the Performance Results section of this paper. This graph shows that as the file size increases a steady state for read access is reached at about 550KB/sec. The test suite was also FTPed back to Osaka by multiple sessions. The read rates were recorded. The results of this test are in Figure 4.4-5 of the Performance Results section of this paper. The results of adding multiple sessions for FTP get access also reached a steady state at around 500KB/sec.

f) A 2GB file was generated to the RAID from Osaka. The creation and deletion time was recorded. This test was run twice. The results of this test can be found in

Figure 4.4-6 of the Performance Results section of this paper. This graph shows that the file generation time for 2GB is about 72KB/sec.

4.3.2 AFS Out-of-Cell Performance Series

Series Purposes:

- Establish performance benchmarks on AFS access from outside of cell
- Demonstrate possible EDOS/ECS method of interaction

Series Config:

- AFS Server on Kyoto
- AFS Client on Kyoto
- AFS Client on Osaka
- ECCS RAID disk used by Kyoto

Series Tests Performed:

a) The test suite was transferred to RAID via FTP from Yakuska to the AFS client on Kyoto. The transfer times were recorded. The results of this test are in Figure 4.4-7 of the Performance Results section of this paper. This graph shows higher transfer rates for smaller files while the transfer rates decrease as file size increases.

b) The test suite was transferred to RAID via FTP from Yakuska to the AFS client on Kyoto. Each file was FTPed back from RAID to Osaka after the original transfer was completed and the file was closed. The transfer rates were recorded. The results of this test are in Figure 4.4-8 of the Performance Results section of this paper. This graph shows that the FTP get times reach a steady state at about 500KB/sec.

c) The test suite was transferred to RAID via FTP from Yakuska to the AFS client on Kyoto. Each file was read back from RAID to Osaka before the file transfer was complete. This test was designed to determine the behavior of file access before file closure. The observed behavior was that for smaller files that do not require multiple writes to disk, the files were not read and a read error occurred. For files that required more than one write to disk, the data that was flushed to disk was read, and there was no read error. There was no way to tell whether or not the entire file was being read or not. There are no performance numbers for this test.

d) The test suite was transferred to RAID via FTP from Yakuska to the AFS client on Kyoto. Each file was FTPed back from RAID to Osaka after the original transfer was completed and the file was closed while the next file was being transferred. The transfer results are in Figure 4.4-9 and the results of the file writes are in Figure 4.4-10 of the Performance Results section of this paper. The get times show a steady state reached around 520KB/sec and the write times decrease as file size increases.

4.3.3 MR-AFS In-Cell Functional/Performance Series

Series Purposes:

- Establish MR-AFS functional/performance benchmarks within cell
- Initial MR-AFS cell administration and setup experience

Series Config:

- MR-AFS Server on Kyoto
- AFS Client on Osaka
- ECCS RAID disk used by Kyoto
- Normal user and administrator accounts

Series Tests Performed:

a) The test suite of files was copied from Osaka to RAID using the Unix cp command. The transfer and deletion times were recorded. The transfer results are in Figure 4.4-11 and the delettion times are in Figure 4.4-12 of the Performance Results section of this paper. The graph shows a rate of about 120KB/sec transfer times.

b) The following MR-AFS rights were toggled and verified for proper user and administrative access to directories. There were no abnormal results.

   - LOOKUP
   - INSERT
   - DELETE
   - ADMINISTER

c) The following MR-AFS rights were toggled and verified for proper user and administrative access to files. There were no abnormal results.

   - READ
   - WRITE
   - LOCK

d) The test suite was generated from Osaka to RAID using a file generation tool. The creation times of theses files were recorded. The results of this test are in Figure 4.4-13 of the Performance Results section of this paper. This graph shows an inconsistent pattern for the small files while large files showed a steady state pattern.

e) The test suite was copied to the RAID device and the files were FTPed back to Osaka. The FTP rates were recorded. The results of this test are in Figure 4.4-14 of the Performance Results section of this paper. This graph shows a ramp up to a steady state at arount 500KB/sec. The test suite was also FTPed back to Osaka by multiple sessions. The FTP rates were recorded. The results of this test are in Figure 4.4-15 of the Performance Results section of this paper. This graph shows a steady state reached at about 470KB/sec with multiple sessions.

f) A 2GB files was generated to RAID from Osaka. The creation time was recorded. This test was run 2 times. The results of this test can be found in Figure 4.4-16 of the Performance Results section of this paper. This graph shows an average creation time of 71.5KB/sec.

4.3.4 MR-AFS Out-Of-Cell Performance Series

Series Purposes:

- Establish performance benchmarks on MR-AFS access from outside of cell
- Demonstrate possible EDOS/ECS method of interaction

Series Config:

- MR-AFS Server on Kyoto
- AFS Client on Kyoto
- AFS Client on Osaka
- ECCS RAID disk used by Kyoto

Scenarios:

a) The test suite was transferred to RAID via FTP from Yakuska to the AFS client on Kyoto. The transfer times were recorded. The results of this test are in Figure 4.4-17 of the Performance Results section of this paper. This graph shows higher transfer rates on the smaller files and at the 1MB file size the rate evens out.

b) The test suite was transferred to RAID via FTP from Yakuska to the AFS client on Kyoto. Each file was FTPed back from RAID to Osaka after the original transfer as completed and the file was closed. The FTP rates were recorded. The results of this test are in Figure 4.4-18 of the Performance Results section of this paper. This graph shows a ramp up to a steady state at about 500KB/sec.

c) The test suite was transferred to RAID via FTP from Yakuska to the AFS client on Kyoto. Each file was read back from RAID to Osaka before the file transfer was complete. This test was designed to determine the behavior of file access before file closure. The observed behavior was that for smaller files that do not require multiple writes to disk, the files were not read and a read error occurred. For files that required more than one write to disk, the data that was flushed to disk was read, and there was no read error. There was no way to tell whether or not the entire was was being read or not. There are no performance numbers for this test.

d) The test suite was transferred to RAID via FTP from Yakuska to the AFS client on Kyoto. Each file was FTPed back from RAID to Osaka after the original transfer was completed and the file was closed while the next file was being transferred. The results of the file reads are in Figure 4.4-19 and the results of the file writes are in Figure 4.4-20 of the Performance Results section of this paper. These graphs show read times to be consistent at about 120KB/sec with concurrent writes and write times be higher with small files and lower with large files while writing is concurrent.

4.4 Performance Results

Each figure of these results represents a complete scenario from the functional/performance series indicated above.

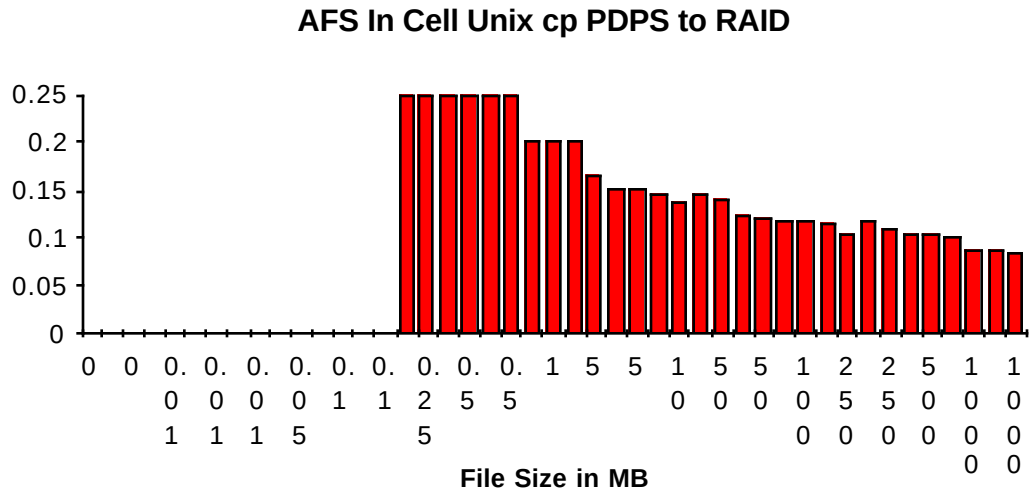

Figure 4.4-1. AFS In Cell Unix cp PDPS to RAID

The transfer rates for the small files did not register in the log file due to system time constraints. The system time call only calculates whole seconds and the transfers were less than a second, therefore they are registered as zero on the graph.

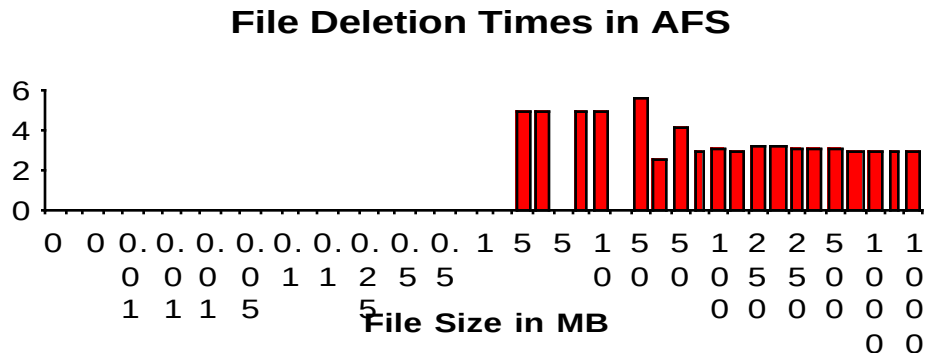

Figure 4.4-2. File Deletion Times in AFS

The deletion times for the small files did not register in the log file due to system time constraints. The system time call only calculates whole seconds and the deletions were less than a second, therefore they are registered as zero on the graph.

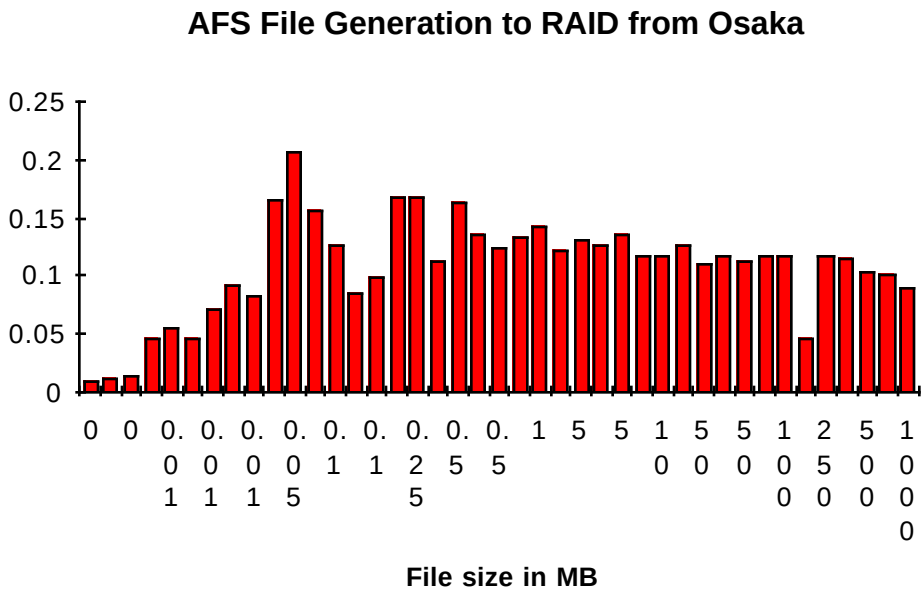

Figure 4.4-3. AFS File Generation to RAID from Osaka

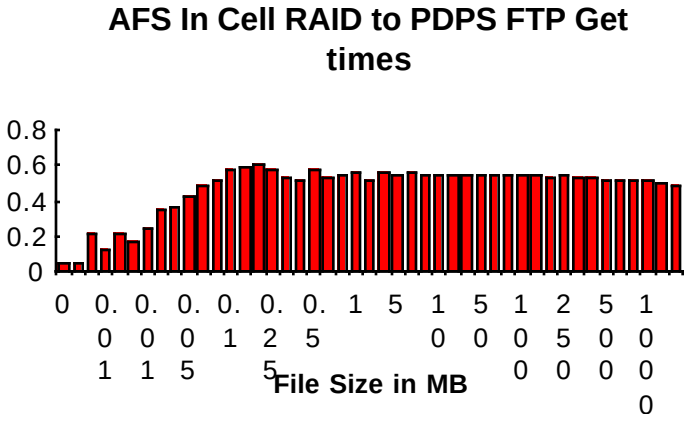

Figure 4.4-4. AFS In Cell RAID to PDPS FTP Get Times

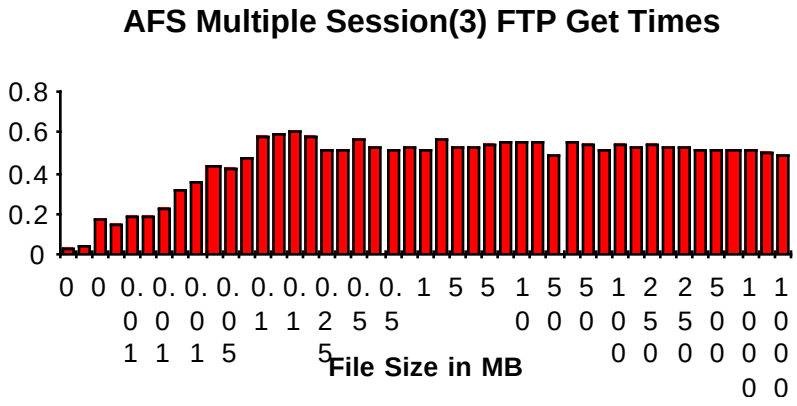

Figure 4.4-5. AFS Multiple Session(3) FTP Get Times

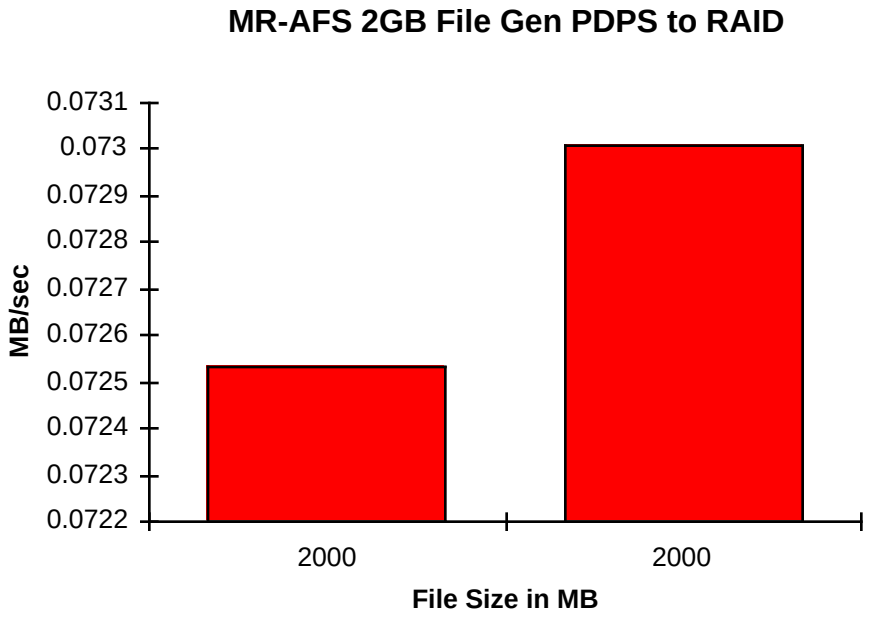

Figure 4.4-6. MR-AFS 2GB File Gen PDPS to RAID

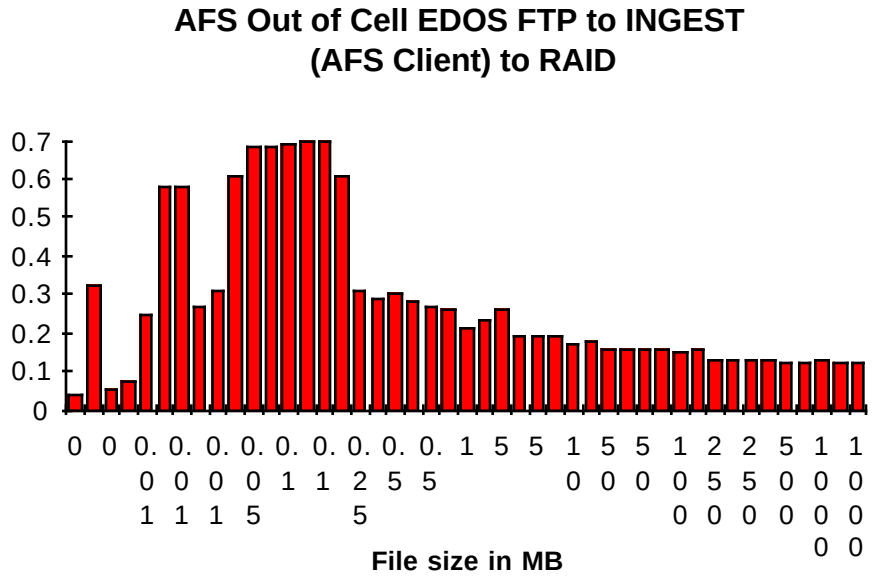

Figure 4.4-7. AFS Out of Cell EDOS FTP to INGEST (AFS Client) to RAID

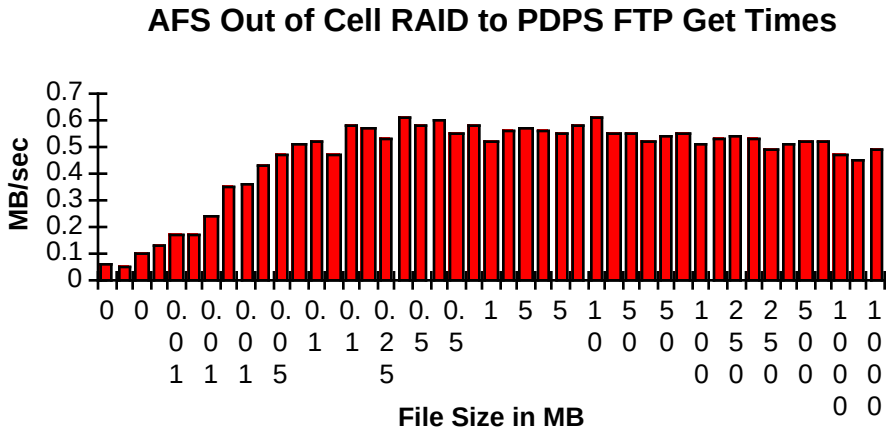

Figure 4.4-8. AFS Out of Cell RAID to PDPS FTP Get Times

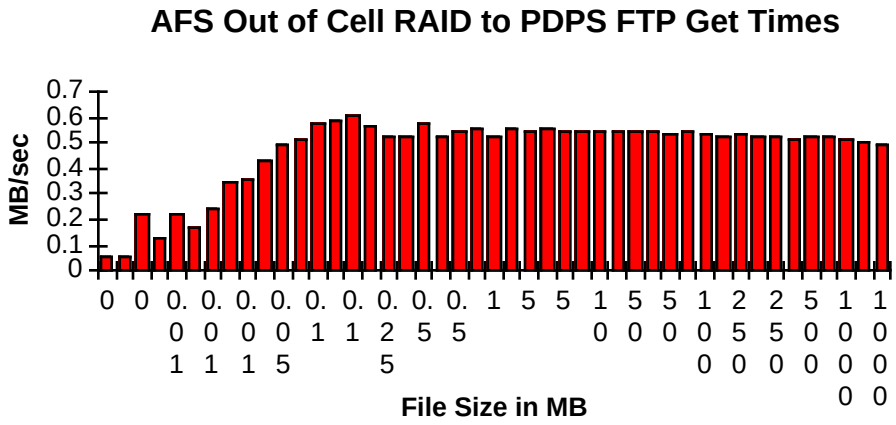

Figure 4.4-9. AFS Out of Cell RAID to PDPS FTP Get Times

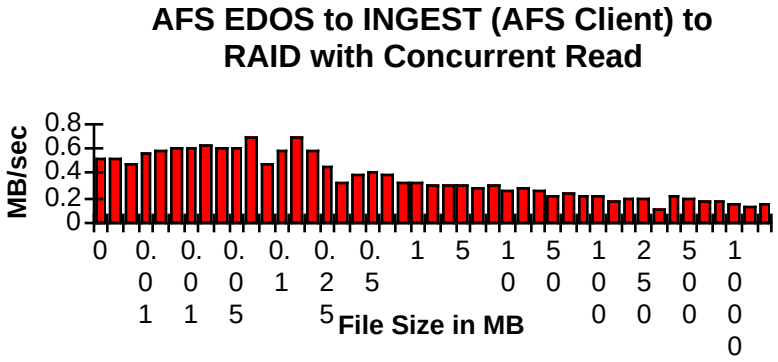

Figure 4.4-10. AFS EDOS to INGEST (AFS Client) to RAID with Concurrent Read

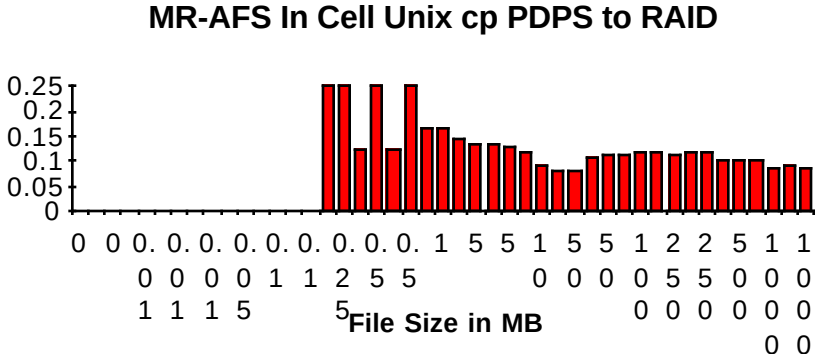

Figure 4.4-11. MR-AFS In Cell Unix cp PDPS to RAID

The transfer rates for the small files did not register in the log file due to system time constraints. The system time call only calculates whole seconds and the transfers were less than a second, therefore they are registered as zero on the graph.

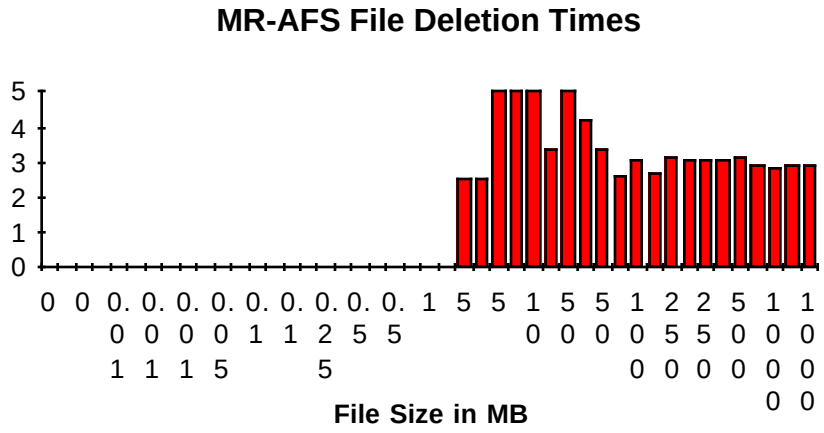

Figure 4.4-12. MR-AFS File Deletion Times

The deletion times for the small files did not register in the log file due to system time constraints. The system time call only calculates whole seconds and the deletions were less than a second, therefore they are registered as zero on the graph.

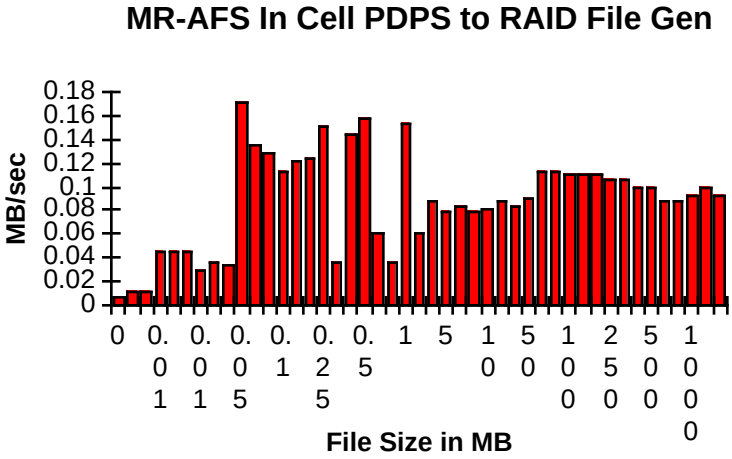

Figure 4.4-13. MR-AFS In Cell PDPS to RAID File Gen

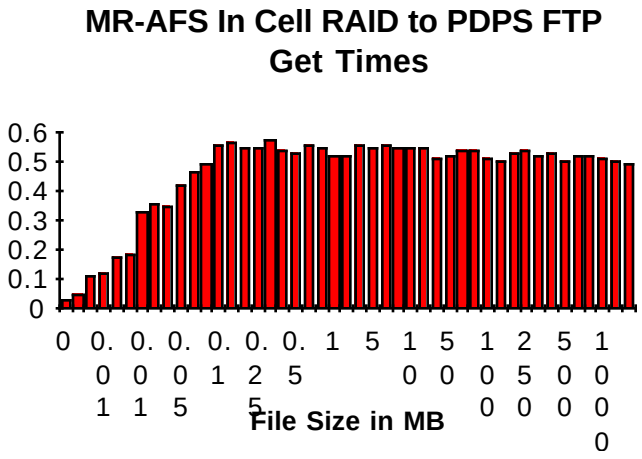

Figure 4.4-14. MR-AFS In Cell RAID to PDPS FTP Get Times

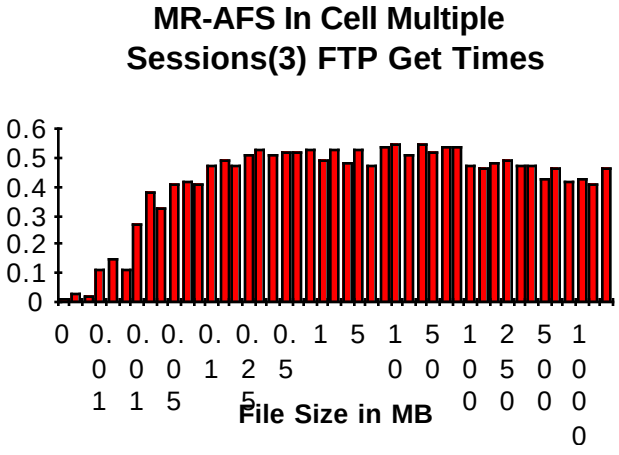

Figure 4.4-15. MR-AFS In Cell Multiple Sessions(3) FTP Get Times

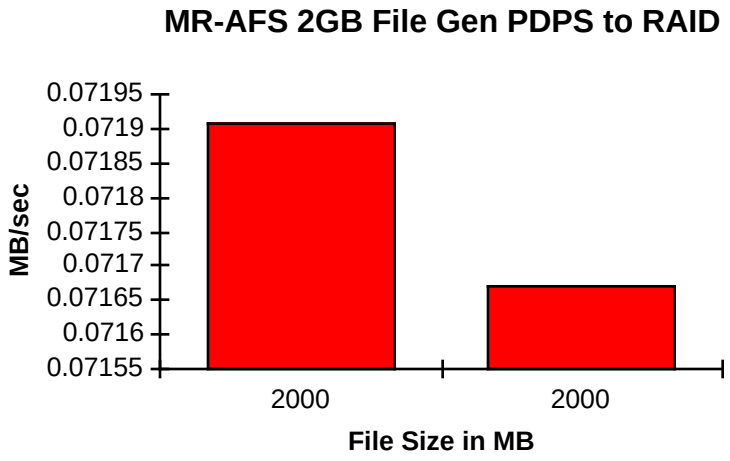

Figure 4.4-16. MR-AFS 2GB File Gen PDPS to RAID

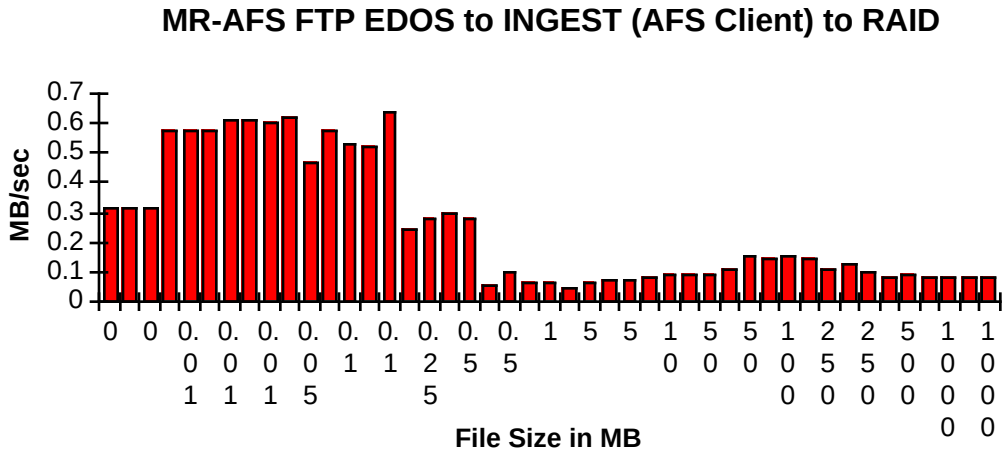

Figure 4.4-17. MR-AFS FTP EDOS to INGEST (AFS Client) to RAID

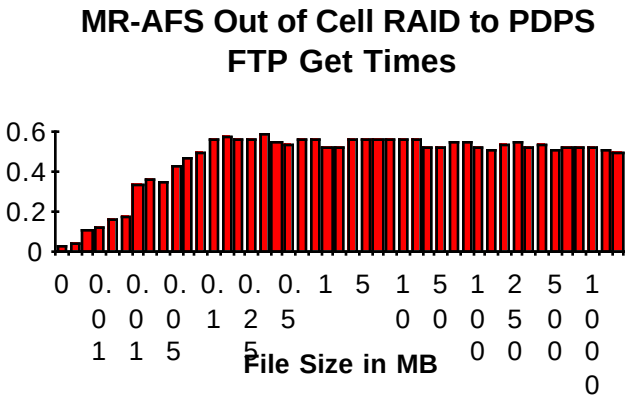

Figure 4.4-18. MR-AFS Out of Cell RAID to PDPS FTP Get Times

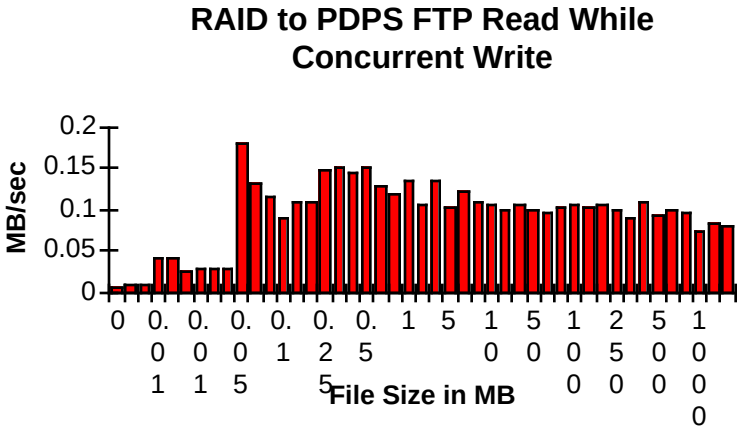

Figure 4.4-19. RAID to PDPS FTP Read While Concurrent Write

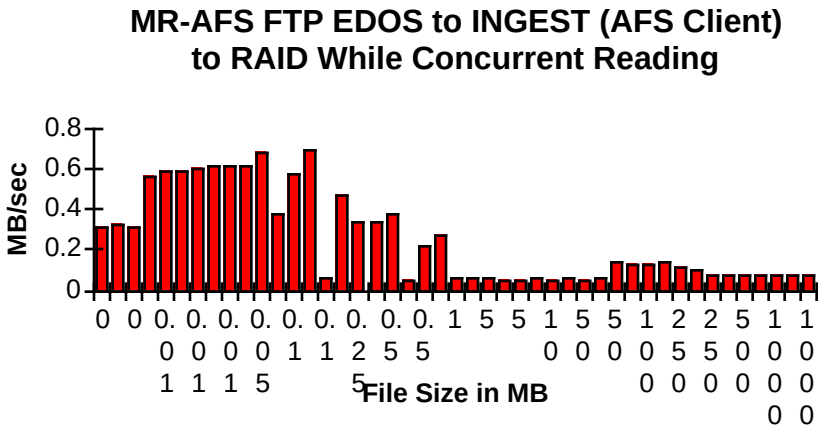

Figure 4.4-20. MR-AFS FTP EDOS to INGEST (AFS Client) to RAID While Concurrent Reading

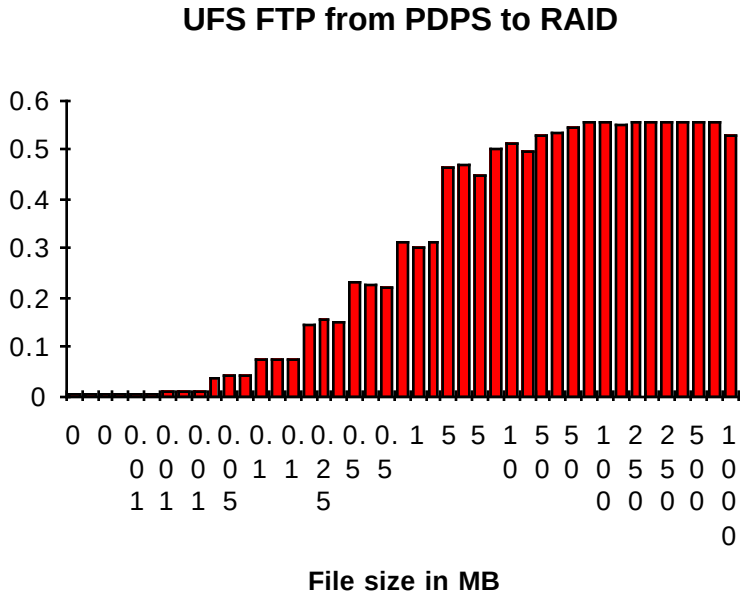

Figure 4.4-21. UFS FTP from PDPS to RAID

4.5 Problems with the Proof of Concept

4.5.1 Hardware Problems

There was a problem during the setup and configuration of the RAID using standard UFS where one machine would be performing read/write tests on the RAID partition and the other machine would be running monitoring software to probe the SCSI bus for statistics. During these initial tests some commands to the SCSI bus would time-out and the bus would perform a hard reset. It would fail with the following error messages:

Nov 4 09:03:49 kyoto unix: WARNING:
/iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000 (esp1):

Nov 4 09:03:49 kyoto unix: Disconnected tagged cmds (1) time-out for Target 0.0

Nov 4 09:03:49 kyoto unix: WARNING:
/iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@0,0
(sd15):

Nov 4 09:03:49 kyoto unix: SCSI transport failed: reason 'time-out': retrying command

Nov 4 09:03:49 kyoto unix: WARNING:
/iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@0,0
(sd15):

Nov 4 09:03:49 kyoto unix: SCSI transport failed: reason 'reset': retrying command

Nov 4 09:03:49 kyoto unix: WARNING:
/iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@0,0
(sd15):

Nov 4 09:03:49 kyoto unix: SCSI transport failed: reason 'reset': giving up

Nov 4 09:03:52 kyoto unix: WARNING:
/iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000 (esp1):

Nov 4 09:03:52 kyoto unix: Identify message 0x80 from Target 0

Nov 4 09:03:52 kyoto unix: WARNING:
/iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000 (esp1):

Nov 4 09:03:52 kyoto unix: Two byte message 'SIMPLE QUEUE TAG' 0xdc rejected

Nov 4 09:03:52 kyoto unix: sd15: incomplete read- retrying

Nov 4 09:06:17 kyoto unix: WARNING:
/iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@0,0
(sd15):

Nov 4 09:06:17 kyoto unix: device busy too long

Nov 4 09:06:17 kyoto unix: WARNING:
/iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@0,0
(sd15):

Nov 4 09:06:17 kyoto unix: SCSI transport failed: reason 'reset': retrying command

Nov 4 09:07:29 kyoto unix: WARNING:
/iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000 (esp1):

Nov 4 09:07:29 kyoto unix: Disconnected tagged cmds (8) time-out for Target 0.0

Nov 4 09:07:29 kyoto unix: WARNING: /iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@0,0 (sd15):

Nov 4 09:07:29 kyoto unix: SCSI transport failed: reason 'time-out': giving up

Nov 4 09:08:40 kyoto unix: WARNING: /iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000 (esp1):

Nov 4 09:08:40 kyoto unix: Disconnected tagged cmds (8) time-out for Target 0.0

Nov 4 09:08:40 kyoto unix: WARNING: /iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@0,0 (sd15):

Nov 4 09:08:40 kyoto unix: SCSI transport failed: reason 'time-out': retrying command

Nov 4 09:08:40 kyoto unix: WARNING: /iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@1,0 (sd16):

Nov 4 09:08:40 kyoto unix: SCSI transport failed: reason 'reset': retrying command

Nov 4 09:08:40 kyoto unix: WARNING: /iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@3,0 (sd18):

Nov 4 09:08:40 kyoto unix: SCSI transport failed: reason 'reset': retrying command

Nov 4 09:08:49 kyoto unix: data transfer overrun

Nov 4 09:08:49 kyoto unix: esp: State=DATA Last State=DATA_DONE

Nov 4 09:08:49 kyoto unix: esp: Latched stat=0x91<IPND,XZERO,IO> intr=0x10<BUS> fifo 0x8a

Nov 4 09:08:49 kyoto unix: esp: last msg out: IDENTIFY; last msg in: DISCONNECT

Nov 4 09:08:49 kyoto unix: esp: DMA csr=0x40040010<INTEN>

Nov 4 09:08:49 kyoto unix: esp: addr=fc0aebc8 dmacnt=0 last=fc0ae9c8 last_cnt=200

Nov 4 09:08:49 kyoto unix: esp: Cmd dump for Target 0 Lun 0:

Nov 4 09:08:49 kyoto unix: esp: cdblen=6, cdb=[0x8 0x3 0x9c 0x3a 0x1 0x0]

Nov 4 09:08:49 kyoto unix: esp: pkt_state=0xb<XFER,SEL,ARB> pkt_flags=0x4000 pkt_statistics=0x2

Nov 4 09:08:49 kyoto unix: esp: cmd_flags=0x10422 cmd_timeout=60

Nov 4 09:08:49 kyoto unix: WARNING: /iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000 (esp1):

Nov 4 09:08:49 kyoto unix: Target 0.0 reducing sync. transfer rate

Nov 4 09:08:49 kyoto unix: polled command time-out

Nov 4 09:08:49 kyoto unix: esp: State=DATA Last State=DATA_DONE

Nov 4 09:08:49 kyoto unix: esp: Latched stat=0x91<IPND,XZERO,IO> intr=0x10<BUS> fifo 0x8a

Nov 4 09:08:49 kyoto unix: esp: last msg out: IDENTIFY; last msg in: DISCONNECT

Nov 4 09:08:49 kyoto unix: esp: DMA csr=0x40040010<INTEN>

Nov 4 09:08:49 kyoto unix: esp: addr=fc0aebc8 dmacnt=0 last=fc0ae9c8 last_cnt=200

Nov 4 09:08:49 kyoto unix: esp: Cmd dump for Target 0 Lun 0:

Nov 4 09:08:49 kyoto unix: esp: cdblen=6, cdb=[0x8 0x3 0x9c 0x3a 0x1 0x0]

Nov 4 09:08:49 kyoto unix: esp: pkt_state=0xb<XFER,SEL,ARB> pkt_flags=0x4001 pkt_statistics=0x2

Nov 4 09:08:49 kyoto unix: esp: cmd_flags=0x10422 cmd_timeout=60

Nov 4 09:08:49 kyoto unix: WARNING: /iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@0,0 (sd15):

Nov 4 09:08:49 kyoto unix: SCSI transport failed: reason 'data_ovr': retrying command

Nov 4 09:08:49 kyoto unix: WARNING: /iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@0,0 (sd15):

Nov 4 09:08:49 kyoto unix: SCSI transport failed: reason 'reset': retrying command

Nov 4 09:08:50 kyoto unix: WARNING: /iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@1,0 (sd16):

Nov 4 09:08:50 kyoto unix: SCSI transport failed: reason 'reset': retrying command

Nov 4 09:08:50 kyoto unix: WARNING: /iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@2,0 (sd17):

Nov 4 09:08:50 kyoto unix: SCSI transport failed: reason 'reset': retrying command

Nov 4 09:08:50 kyoto unix: WARNING: /iommu@f,e0000000/sbus@f,e0001000/dma@1,81000/esp@1,80000/sd@3,0 (sd18):

Nov 4 09:08:50 kyoto unix: SCSI transport failed: reason 'reset': retrying command

There were two items that were associated with the temporary solution and workaround to this problem. The first item identified was the SCSI identification number for each machine had the same value. The other item has to do with the way the SCSI protocol queues commands. The ECCS technical people found that there was an error in their controller firmware. This error was causing the time-out problem. The workaround to this problem was to turn off SCSI tagged command queuing so that one and only one SCSI command at a time would be serviced. This caused a slight degradation in performance under ECCS SCSI level performance analysis. This change offered a temporary solution to the problem. ECCS has since fixed the problem in their next generation of RAID device and will go back and fix the problem in this model. These problems were found and fixed during installation and configuration of the RAID device and before the AFS software was installed therefore not affecting functionality.

During initial configuration and installation it was noticed that the ECCS RAID device was inherently slow. Transfer rates were not as claimed and not very good. The performance numbers for standard UFS file copies can be seen in Figure 4.4-21 of the Performance Results section of

this paper and should be used only in comparison to the performance numbers for AFS and MR-AFS.

There was an anomaly noticed during Out of Cell testing of both AFS and MR-AFS where files that were FTPed from Yakuska to Kyoto would lose 1 byte for every 1KB of file transferred. This problem has not been resolved as of the writing of this paper, but technical support for MR-AFS has stated that this problem has been seen before and had been traced to the router configuration and the FTP packet size. Investigations by Bruce Clark revealed that this problem was due to a "bug" in the MR-AFS *ftp daemon*. During default (ASCII) transfers, bytes were lost. When *ftp* was explicitly placed into binary mode, no data was lost. This information was provided to PSC for problem resolution.

This page intentionally left blank.

5. Conclusions

From the goals set forth in this paper it can be concluded that AFS is a viable solution in meeting SDPS requirements in its functional capabilities. The functional testing of access control lists in determining authorized and unauthorized users was successful and complete. It was shown in this testing that access from multiple points to the same data is transparent and without loss of continuity. AFS is capable of handling any size files up to 2GB with only minor variance from the performance experienced with UFS.

MR-AFS is also viable solution in meeting SDPS requirements. It was determined through these tests that MR-AFS was not defective in function or performance when put through the same series of tests as AFS. MR-AFS will allow multiple hosts to simultaneously share for reading and writing a large "free pool" of RAID space without data loss or significant performance degradation.

The performance of the system as configured was less than desired. Since the main thrust of this effort was the functional ability of both AFS and MR-AFS the system suffered from a lack of optimum tuning. The were many parameters of the RAID device that were tunable but were not tuned. The idea being to get a stable system and keep it constant throughout the entire PCON and focus on the qualitative aspects of this effort. In that light, the performance numbers that were collected should be used as a relative guide between the performance of AFS versus MR-AFS in determining any degradation of performance due to the multi-resident extensions of MR-AFS. If this would have been a performance based effort then there would have been more data points collected.

While neither of these products represents an entire solution to the challenges that are present in quest for a truly network attached storage solution, both represent building blocks with which the SDPS requirements can be attacked and conquered.

This page intentionally left blank.

6. References

AFS Command Reference Manual, Transarc Corporation, 1993.

AFS User's Guide, Transarc Corporation, 1993.

AFS System Administrator's Guide, Transarc Corporation, 1993.

MR-AFS System Administration Guide, Pittsburgh Super Computer Advanced File Systems Group, 1994.

This page intentionally left blank.

Appendix A - Prototype Data Points

For brevity the data collected for Prototype 3 will be available as a separate document in the ECS Library and the World Wide Web. ***File name will be inserted here.***

This page intentionally left blank.

Abbreviations and Acronyms

AFS	Andrew File System
ECS	EOSDIS Core System
FDDI	Fiber Distributed Data Interface
FTP	File Transfer Protocol
GB	Giga Byte
KB	Kilo Byte
MB	Mega Byte
NFS	Network File System
OS	Operating System
PCON	Proof of Concept
RAID	Redundant Array of Inexpensive Disks
SCSI	Small Computer System Interface
SDPS	Science Data Processing Segment
UFS	Unix File System
SUN	Sun MicroSystems